Option 2: Extension of Binary SVM into Multiclass Recognition

Data Set: We have two datasets.

(1) **Heart**: This is a 2-category dataset included in the file 'HeartDataSet.mat' in the folder of 'data\'. The file contains 4 matrices, i.e. ,'Xtrain','Ytrain','Xtest','Ytest', where 'Xtrain'/'Ytrain' are the feature/label matrix of the training set, and 'Xtest'/'Ytest' are those of the testing set. Each sample corresponds to a row of the feature/label matrix. Note: use MATLAB command load to read a .mat file into your program.

(2) **Vehicle**: The data set consists of 4 categories, i.e. 'bus', 'saab', 'opel', and 'van'. There are totally 940 samples distributed in the 9 files from 'xaa.dat' to 'xai.dat' in the folder of 'data\'. Each file contains a matrix whose row vector contains the 18-dimensional feature and the category in the last column of each sample. We use the first 7 files as **training set**, i.e. **'xaa.dat' to 'xag.dat'**, and the last 2, i.e. **'xah.dat' and 'xai.dat',** as **testing set**.

Step 0: Binary SVM

Output: the testing accuracy on 'Xtest'/'Ytest' of the Heart dataset.

Instructions: Implementation of binary SVM (in its primal form) by Quadratic Programming. A **quadratic programming** (**QP**) problem is:

$$\begin{aligned} &\min_{\mathbf{z}} \mathbf{z}^T \cdot \boldsymbol{Q} \cdot \mathbf{z} + \boldsymbol{c}^T \cdot \boldsymbol{z} \\ &s.t.\ \boldsymbol{A} \cdot \mathbf{z} \le \mathbf{g},\ \ \mathbf{E} \cdot \mathbf{z} = \mathbf{d} \end{aligned} \tag{1}$$

The primal cost function of SVM is:

$$\begin{aligned} &\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &s.t.\ y_i \cdot [(\mathbf{w})^T\mathbf{x}_i + b] \ge 1 - \xi_i, 1 \le i \le N, \\ &\xi_i \ge 0, \quad 1 \le i \le N \end{aligned} \tag{2}$$

where $\mathbf{x}_i \in R^M$ is the feature vector and y_i is the label of the i-th sample ($1 \le i \le N$). The SVM problem (2) can be converted into a QP problem (1) as follows. First, set the $\boldsymbol{z}$, $\boldsymbol{Q}$, and $\boldsymbol{c}$ in (1) as:

$$\boldsymbol{z} = \begin{bmatrix} \mathbf{w} \\ b \\ \boldsymbol{\xi} \end{bmatrix},\ \boldsymbol{Q} = \begin{bmatrix} \boldsymbol{I}_{M\times M} & \mathbf{0}_{M\times(N+1)} \\ \mathbf{0}_{(N+1)\times M} & \mathbf{0}_{(N+1)\times(N+1)} \end{bmatrix}, \boldsymbol{c} = \begin{bmatrix} \mathbf{0}_{(M+1)\times 1} \\ C\cdot\mathbf{1}_{N\times 1} \end{bmatrix},$$

where $\boldsymbol{\xi} = [\xi_1, \dots, \xi_N]^T$, $\boldsymbol{I}_{M\times M}$ is an $M \times M$ diagonal matrix, $\mathbf{0}_{a\times b}$ and $\mathbf{1}_{a\times b}$ are an $a \times b$ matix of all 0's or 1's. Second, the matrix $\boldsymbol{A}$ in (1) will be $(2N) \times (N + M + 1)$, whose row $1{\sim}N$ is:

$$\boldsymbol{A}(i,:) = -\left[y_i \cdot \mathbf{x}_i{}^T,\ 1,\ \mathbf{0}_{1\times(i-1)}, 1,\ \mathbf{0}_{1\times(N-i)}\right], 1 \le i \le N\,.$$

The remaining row $(N + 1){\sim}2N$is:

$$\boldsymbol{A}(N+i,:) = -\left[\mathbf{0}_{1\times(M+1)},\ \mathbf{0}_{1\times(i-1)},\ 1,\ \mathbf{0}_{1\times(N-i)}\right].$$

The corresponding $\mathbf{g}$ in (1) is $\mathbf{g} = [\mathbf{1}_{1\times N},\ \mathbf{0}_{1\times N}]^T$**.** Finally, since we have no equality constraints in the SVM primal, $\boldsymbol{E}$ and $\mathbf{d}$ can be considered to be 0.

NOTE: The Matlab function of Quadratic programming is : `quadprog`.

Step 1: One-vs-all extension.

Output: Implement the one-vs-all extension of binary SVM for multi-class recognition. Train the SVM by the training set, and report the accuracy in the testing set of the 'Vehicle' dataset.

Instructions: Suppose we have K sample categories. Take the k-th category out as positive and leave the remaining $(K-1)$ categories as negative, and we can train an SVM classifier $h_k(\mathbf{x}) = (\mathbf{w}_k)^T\mathbf{x} + b_k$ for the k-th category. The final classification result is:

$$k^* = \text{argmax}_k[(\mathbf{w}_k)^T\mathbf{x} + b_k].$$

NOTE: More details of one-vs-all extension can be find in any pattern recognition book, e.g. Section 3.7.3 in [1].

Step 2: One-vs-one extension

Output: Implement the one-vs-one extension of binary SVM for multi-class recognition. Train the SVM by the training set, and report the accuracy in the testing set of the 'Vehicle' dataset.

Instructions: Suppose we have K sample categories. Build $K(K-1)/2$ SVM classifiers where one classifier $h_{k,j}(\mathbf{x}) = (\mathbf{w}_{k,j})^T\mathbf{x} + b_{k,j}$ distinguishes a pair of categories k and j by taking k as positive and j as negative. With $h_{k,j}(\mathbf{x}) = -h_{k,i}(\mathbf{x})$, the final classification is the majority voting process below:

$$k^* = \text{argmax}_k \sum_j \text{sign}[h_{k,j}(\mathbf{x})].$$

NOTE: More details of one-vs-one extension can be find in any pattern recognition book, e.g. Section 3.7.3 in [1].

Step 3: **Extension by error-correcting-coding**

Output: Implement the error-correcting-coding extension of binary SVM for multi-class recognition. Train the SVM by the training set, and report the accuracy in the testing set of the 'Vehicle' dataset.

Instruction: Suppose we have K categories, and each category is represented by a binary code word of length L. This is equivalent to constructing a $K \times L$ **coding matrix**. For the 4-category "Vehicle" dataset, we use the coding matrix below.

$$\mathbf{C} = \begin{matrix}\text{bus:}\\ \text{saab:}\\ \text{opel:}\\ \text{van:}\end{matrix}\begin{bmatrix}+1 & -1 & -1 & -1 & +1 & +1 & +1\\ -1 & +1 & -1 & -1 & +1 & -1 & -1\\ -1 & -1 & +1 & -1 & -1 & +1 & -1\\ -1 & -1 & -1 & +1 & -1 & -1 & +1\end{bmatrix}$$

During the **training stage**, for each of the L columns in the coding matrix, we divide the K categories into 2 parts, i.e. $+1$ and -1, according to their code and train a binary classifier $g_l(\mathbf{x}) = \{+1, -1\}, 1 \le l \le L$. For example, the 5-th column of the above coding matrix $\mathbf{C}$ implies 'bus' and 'saab' will be taken as positive, and 'opel' and 'van' are negative.

In the **recognition stage**, a sample $\mathbf{x}'$ will leads to an L-bit code: $\mathbf{c}(\mathbf{x}') = [g_1(\mathbf{x}'),\ g_2(\mathbf{x}'),\ \cdots, g_L(\mathbf{x}')]$ by the L binary classifiers. The final recognition result is $k^* = \text{argmin}_k dist(\mathbf{C}(k,:), \mathbf{c}(\mathbf{x}'))$, where $\mathbf{C}(k,:)$ is the k-th row of coding matrix as the code of k-th category, and $dist(\cdot,\cdot)$ can be any distance metric. In this experiment, we use the Euclid distance, i.e. $dist(\mathbf{x}, \mathbf{y}) = \sqrt[2]{\sum_i [\mathbf{x}(i) - \mathbf{y}(i)]^2}$.

NOTE: More details of **error-correcting-coding** extension can be found in any pattern recognition book, e.g. Section 3.7.3 in [1].

Step 4: (Optional) (Direct formulation of multi-class SVM)

(Optional: You will not be penalized if you do not do this part. Doing this part is helpful for understanding SVM better.)

Output: Implement the multi-class SVM for multi-class recognition. Train the SVM by the training set, and report the accuracy in the testing set of 'Vehicle' dataset.

Instruction: Consider the following direct formulation of K-class SVM [2].

$$\begin{gathered} \min_{\mathbf{w}_k,\xi_i} \frac{1}{2}\sum_{k=1}^{K}\|\mathbf{w}_k\|^2 + C\sum_{i=1}^{N}\xi_i \\ s.t.\ \mathbf{w}_{y_i}{}^T \cdot \mathbf{x}_i - \mathbf{w}_k{}^T \cdot \mathbf{x}_i \geq \Delta(k, y_i) - \xi_i\ , 1 \leq k \leq K, 1 \leq i \leq N, \\ \xi_i\ \geq 0, \quad 1 \leq i \leq N \end{gathered} \tag{3}$$

where $\mathbf{x}_i$ and y_i is the same as (1), and the $\mathbf{w}_k$ is the weight vector of the k-th class ($1 \leq k \leq K$). The final recognition is:

$$k^* = \text{argmax}_k[(\mathbf{w}_k)^T\mathbf{x}].$$

There may be a way of implementing a solution by Quadratic Programming as the direct extension of Step 0. Try to find the solution yourself.

Reference

[1] S. Theodoridis and K. Koutroumbas, Pattern Recognition (4th Edition), Academic Press, 2008.

[2] K. Crammer and Y. Singer. On the Algorithmic Implementation of Multi-class SVMs, JMLR, 2001

Some general requirements for all options:

1. In your project report, please include a section that specifies the respective contributions of each individual group member.
2. While your report should briefly summarize what you did, you do not need to repeat the detailed procedures/equations etc. that are already described in the project descriptions (or papers, for the options based on given papers).
3. Your report should clearly document all major outcomes/results of your project, e.g., plots, error rates, sample images, etc. While the graders may run your code to verify your work, they are not supposed to run your code to obtain the results for grading. (Therefore, you should never say things like "To get the required plots, please run MyCode1.m"; Instead, just include the plots in your report.)
4. Your report may discuss anything interesting you found in the process of doing your project. This is the opportunity for showing your understanding of the problem beyond mere implementation of some given algorithms.
5. Your report should also serve as a manual for running your code to verify your results.
6. Do NOT embed any code in your report. Submit your report, your programing files or data files if any as a single zip file.